



SnapShotBot—Program It

You might think that programming the SnapShotBot is a little more involved than the previous bots. Although the program might be a little larger in size, the truth is that you're already experienced with all the programming blocks you'll need.

For this chapter, I'm going to walk you through the programming using a slightly different method. When I program my bots, many times I create the program in full and then upload it to my bot for testing. From there, I add and remove blocks as needed. Although this is perfectly fine for many bots, for a large program this might not be the best way to test your bot. If you find a mistake early in the program, it can cause you to have to delete other portions of your program. And if you find a *really huge* mistake, you might find yourself deleting the program completely and having to start over.

So, let me show you another method for programming that involves building your program in small steps, uploading the program, then testing it. When you're done with this chapter, you should be able to decide for yourself when it might be beneficial to program in small steps or simply program the entire thing and then test and debug.

One Block at a Time

Get your SnapShotBot Design Journal page and open up the Lego Mindstorms NXT software. Type **SnapShotBot** into the blank text field labeled **Start New Program**, then click the **Go** button (see Figure 12-1).

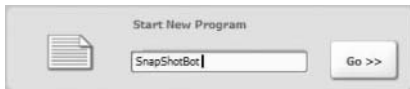


Figure 12-1. Enter *SnapShotBot* for the new program name and click *Go*.

Note To have more work space visible on your screen, close down the RoboCenter area on the far right by clicking the small red X in the upper-right corner of the software.

In the SnapShotBot building instructions in Chapter 11, you might have noticed I placed the Touch Sensor on the back of the bot. I plan on using the Touch Sensor as a sort-of **Start**

button for my bot to get rolling. So, the first item you'll place in your program will be a simple LOOP block (see Figure 12-2) that waits for the Touch Sensor to be triggered (pressed and then released). Once the sensor is triggered, the remaining program will begin. (You could use a WAIT block that breaks when the Touch Sensor is pressed, but I like to use the LOOP block because I can later add blocks inside the LOOP if I want the bot to perform some other actions while it's waiting.)

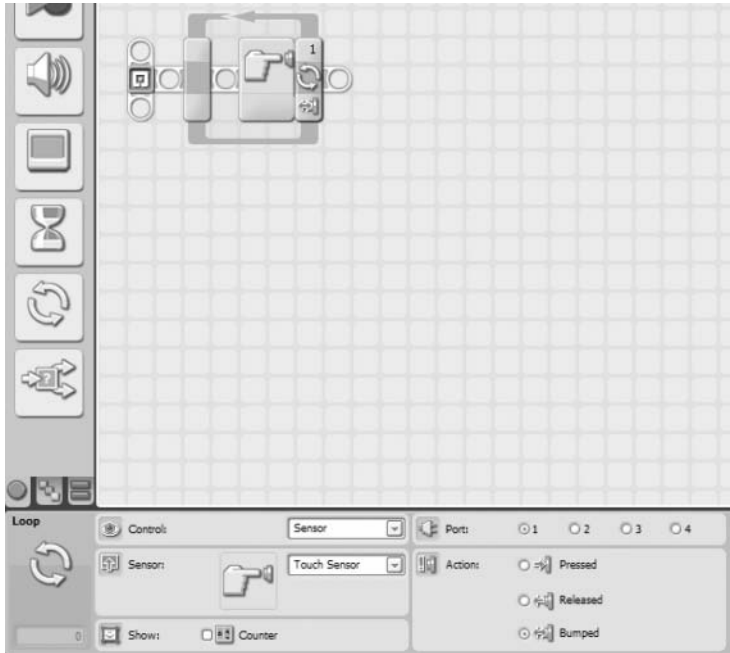


Figure 12-2. Use a LOOP block and Touch Sensor trigger to start the bot.

Now you can start on the actual programming blocks needed for the bot to perform the actions in the Task List. Look at the Task List on your Design Journal page (turn back to Figure 10-3). Our first task is “Move forward to the center of room and stop.” If you’ll think back to Chapter 10, we determined that to get halfway across the room, we needed to program our motors to spin for ten rotations. Each rotation moves the bot approximately seven inches, so when the motors spin for ten rotations, the bot will move forward approximately seventy inches, or almost six feet. That’ll be good enough to get the bot near the center of the room.

Place a MOVE block and configure it with a **Duration** of ten rotations for motors B and C. Be aware that I’ve selected the direction for motors B and C as the down arrow (see Figure 12-3). This is because the motors are reversed on my design—facing away from the back of the bot. Because of this, forward motion for the bot means having these motors spin in the “reverse” direction. If your bot differs, configure your motor directions based on your own design.

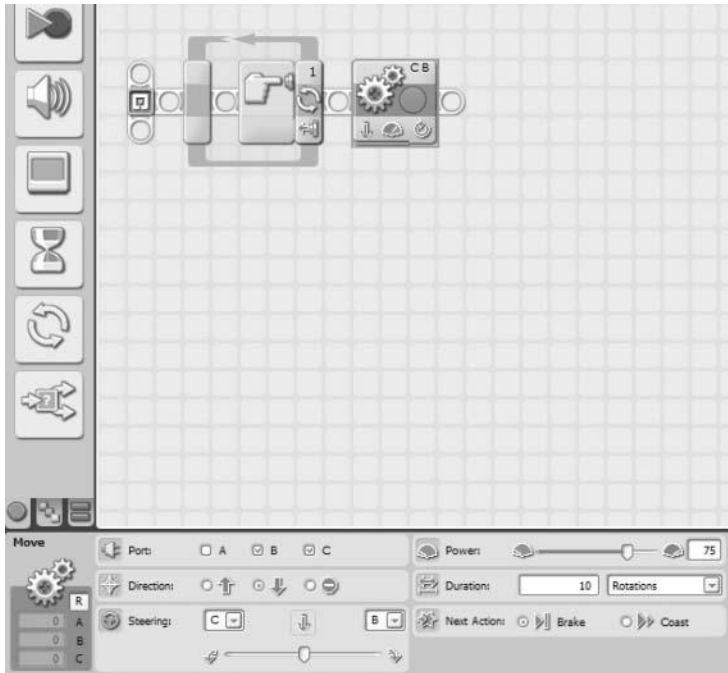


Figure 12-3. Configure the *MOVE* block to move the bot forward, halfway into the room.

Now it's time to test. I'm going to save the program (so I don't lose any of my work or the comments I've added) and upload it to my bot. When I run the program, the bot should wait until I press and release the Touch Sensor. After I trigger the sensor, the bot should move forward about six feet and stop. Here goes the test . . . and it worked, exactly as designed. (If your bot did anything differently, check your programming blocks and verify that the number of rotations is correct.)

Next on our Task List is "Turn Left to face library." Back in Chapter 10, I reminded you how to test your bot and obtain the number of degrees to turn for making a right and left turn. When I tested my bot, I obtained a value of -464 degrees when *only* the right-side wheel was turned (motor B). Some of you might have discovered the software does *not* allow you to enter a negative value for the duration. Don't let this concern you. The value was negative (–) to indicate the motor was turning in reverse. So what you'll do is configure the motor to spin in reverse (the down arrow) for 464 degrees—simple! Drop in your *MOVE* block and configure it as shown in Figure 12-4.

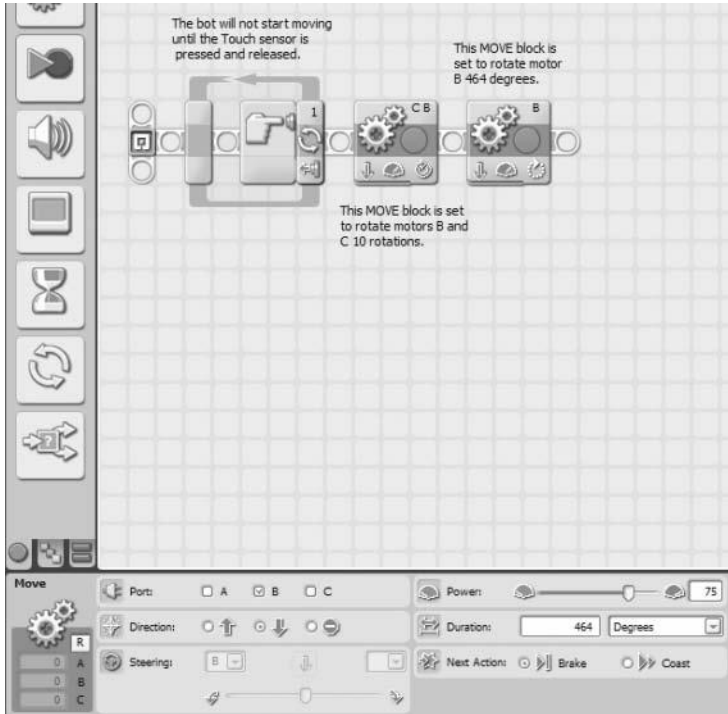


Figure 12-4. This *MOVE* block will make the bot turn left.

Now we test again. When I trigger the Touch Sensor, the bot moves forward about six feet. It then makes a left turn when motor B turns 464 degrees. So far, so good.

The Task List shows our next step is “Take picture.” Before I configure motor A, however, I do want to add in a WAIT block. I’m worried that motor A might push the button on the camera just as the bot is coming to a stop—this would result in a blurred picture. I’ll configure the WAIT block for 2 seconds (see Figure 12-5).

Next we need to configure motor A to turn and press the camera button. Because I don’t know how far to turn it, this is where testing comes in. I dropped in a MOVE block and configured motor A to turn 45 degrees. During testing, I found this sometimes missed the button entirely. I reduced it to 20 degrees and tested again—same result. For my third test, I placed the arm directly on the button of the camera and configured the MOVE block for 10 degrees—it worked! The button pressed and a picture was taken (see Figure 12-6).

Note Depending on the type of camera you’re using, you’ll probably have to perform a few different tests to determine the best way to press the camera button. When trying different motor settings, such as number of rotations or degrees, it’s usually best to start big. Start with a large number of rotations or degrees—if it works, reduce it a little and try again. Keep reducing until you find the lowest setting that works. Your batteries will last much longer!

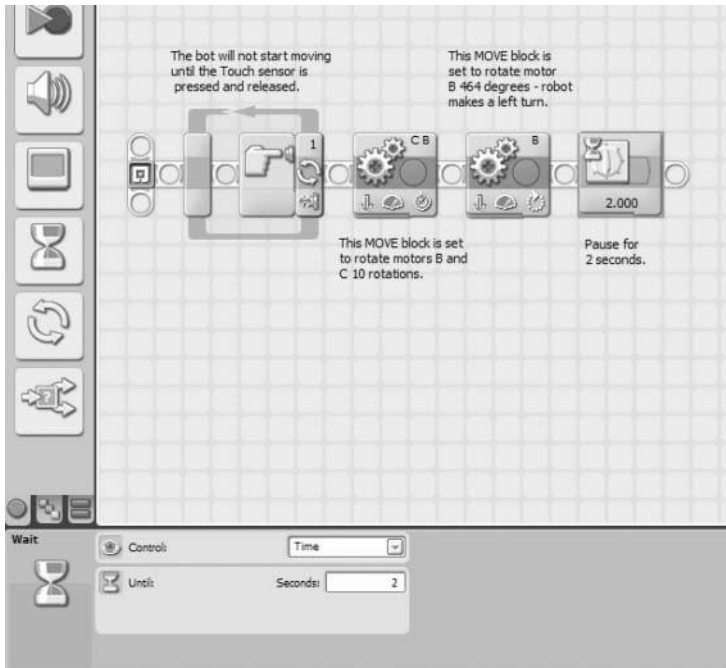


Figure 12-5. A WAIT block allows the bot to stop moving before the picture is taken.

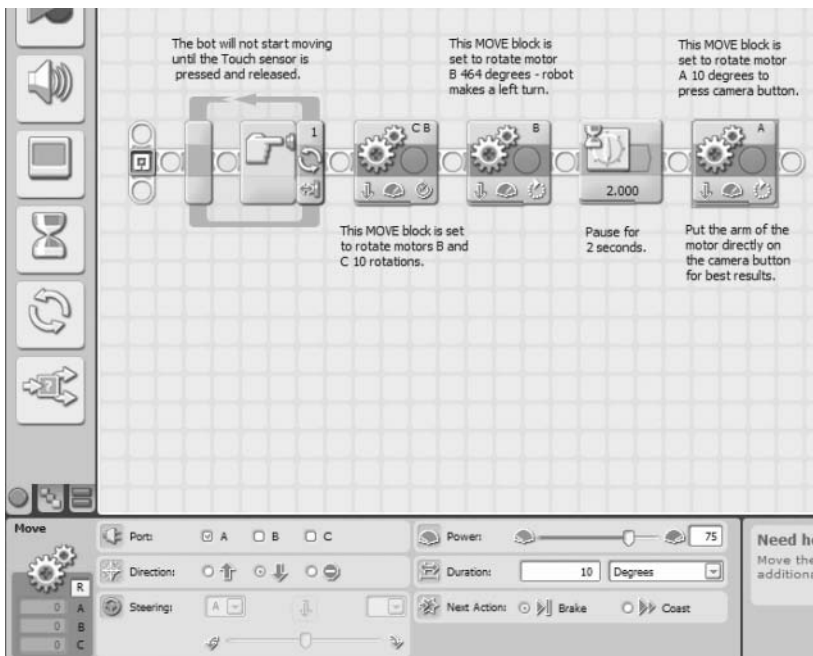


Figure 12-6. Using a MOVE block to take a photograph

Finding the Basket

Okay, at this point, our little bot has made it to the center of the room, turned left, paused, and then taken the picture. Looking at the Task List, I see that my next step is “Move forward until black obsidian ring is detected and stop.” I want my bot to start moving forward until its Light Sensor detects the black obsidian rock surrounding the basket. I also want the bot to move forward slowly so it doesn’t get too close to the basket. When the Light Sensor detects a change in the reflected light, I want the bot to stop.

Back in Chapter 10, I told you how to test your Light Sensor to obtain a reading for a “normal surface” and an “obsidian surface.” When I tested my “normal floor” (a light-colored wood floor), the Light Sensor returned a value of 25. When I placed the dark paper (black) under the Light Sensor, I got a value of 10. Your values will probably differ a bit. I’m going to program my Light Sensor to check for a value of 15 or less to be safe. When the Light Sensor is triggered (value of 15 or less), the bot will stop moving forward because it has detected the “obsidian surface.”

Now, to keep the bot moving forward until the Light Sensor is triggered, you’re going to need to use a LOOP block configured to test the Light Sensor. You can see the settings for the LOOP block in Figure 12-7.

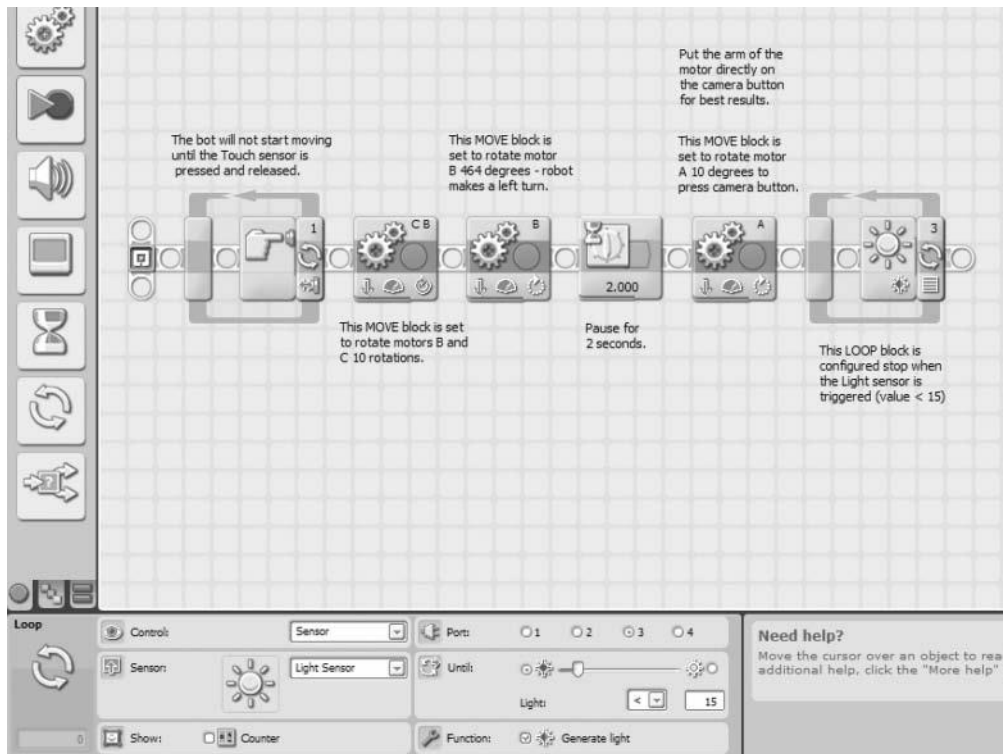


Figure 12-7. Using a LOOP block to test whether the Light Sensor is over the obsidian surface

To make the bot actually approach the basket, throw in a MOVE block. You'll configure this block a little differently than other LOOP blocks you configured for the earlier bots. With this MOVE block, instead of configuring motors B and C with a **Duration** of Unlimited, you'll have them move forward for one rotation of the wheels. After each rotation, the Light Sensor will check to see if it's over the "obsidian surface." If it isn't, the bot will move forward another rotation of the wheels, and the Light Sensor will check again. By doing it this way, you can make sure the bot doesn't rush forward quickly and cross too far over the obsidian ring (see Figure 12-8).

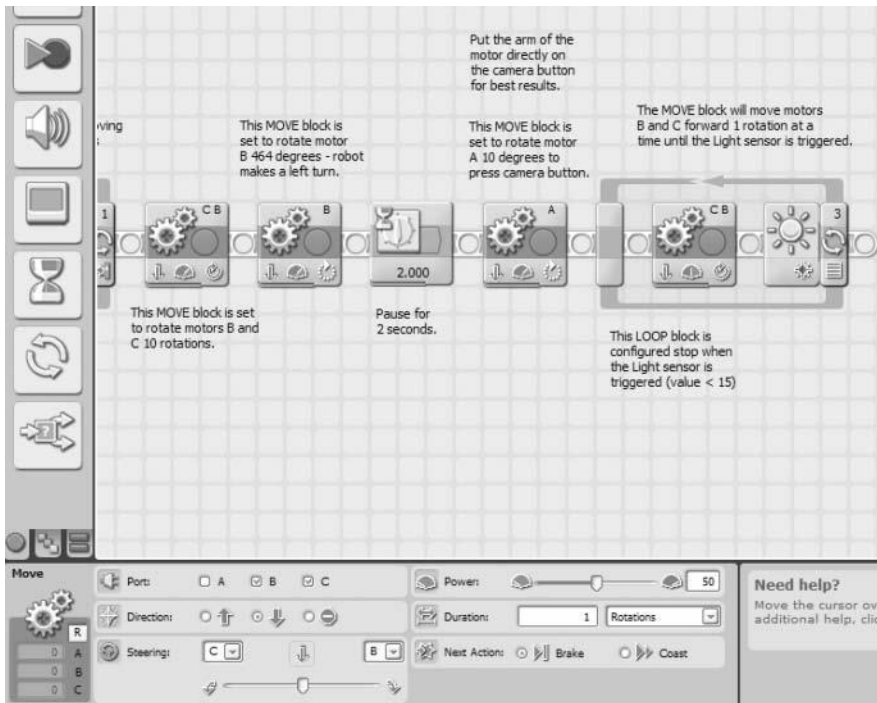


Figure 12-8. Using a LOOP and MOVE block with the Light Sensor

After the Light Sensor is triggered and the bot stops, the bot will need to go around the basket with the twine and prepare for the trip back.

Getting Around the Basket

Now the bot is in front of the basket. What's next? Look at the Task List: "Turn right, move forward short distance and stop."

Our first task will be to get the bot to turn right. Back in Chapter 10, I recorded the value as -460 degrees for a right turn. Remember, the negative sign (–) indicated that motor C was turning in reverse. So you'll insert a MOVE block that's configured to turn motor C in reverse for 460 degrees (see Figure 12-9).



If you examine the mini-map in Figure 10-4, you'll notice that once the bot reaches Point A, it will make three left turns before heading back to its starting point. It will turn left (1) at Point A, move forward, and stop at Point B; turn left (2), move forward, and stop at Point C; turn left (3), then finally move forward, and stop at Point D. This is a perfect location to use a LOOP again. It allows us to reduce the number of programming blocks by performing the same actions three times. Those actions are "Turn Left and Move Forward a Short Distance." So let's place the LOOP first (see Figure 12-11).

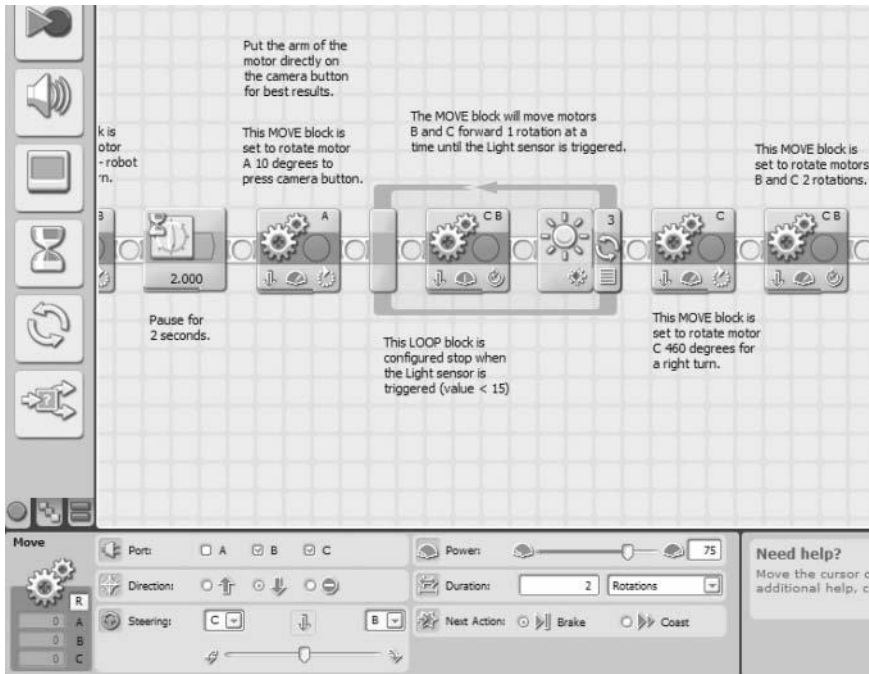


Figure 12-10. Now the motors need to stop.

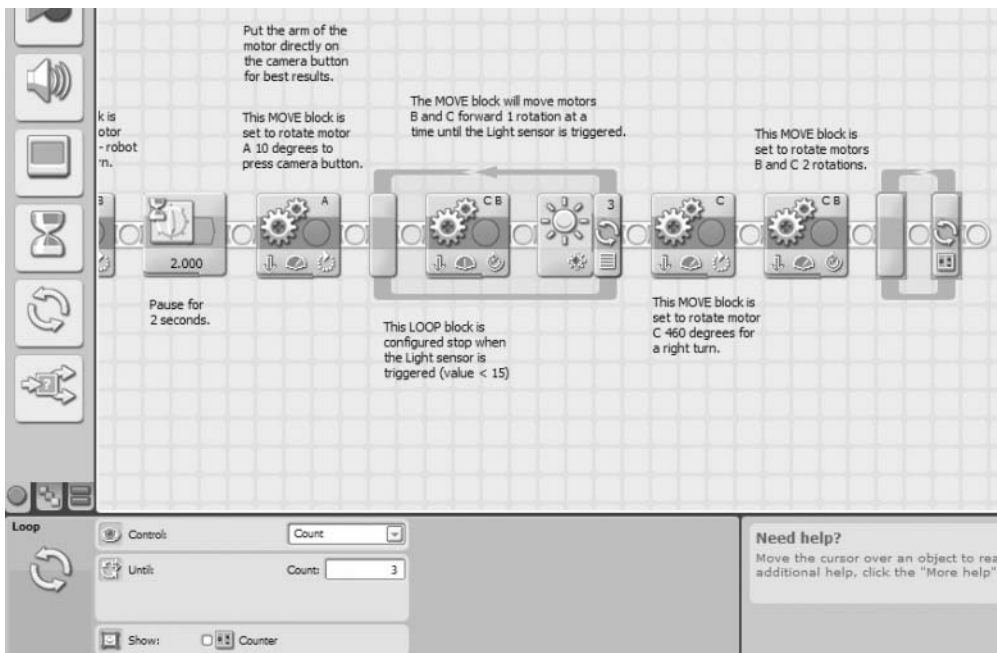


Figure 12-11. The LOOP block gets the bot around the basket.

This book was purchased by jleahy@schools.nyc.gov

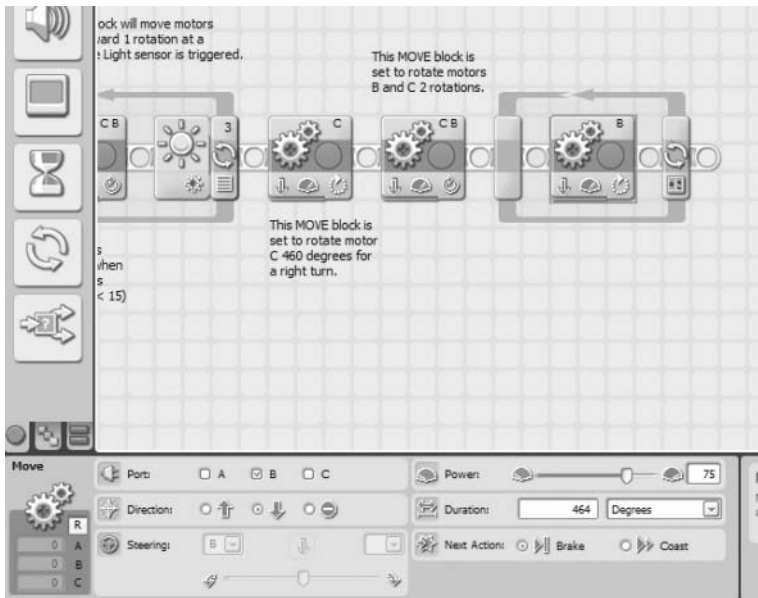


Figure 12-12. *This MOVE block turns the bot left.*

Next, place the MOVE block to move the bot forward 28 inches (see Figure 12-13).

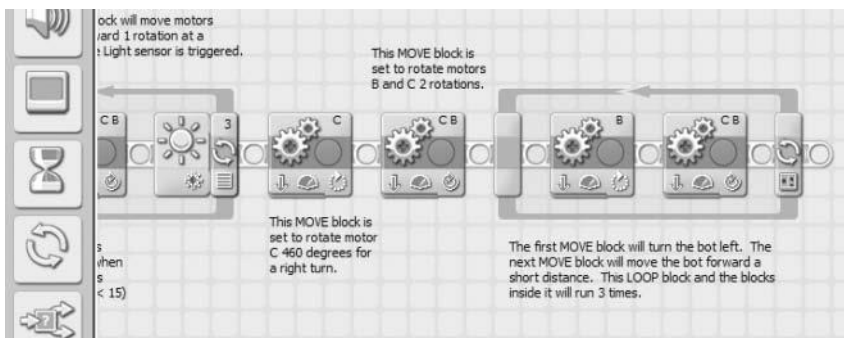


Figure 12-13. This *MOVE* block moves the bot forward a short distance.

Now you need to test your bot. At this point, let me describe my test environment and show you how to set up your own.

First, I'm running this test in my living room. I've cleared away a chair so I'll have plenty of room for my bot to run. In Figure 12-14, you can see how I've set up my test run. I've placed the "basket" (actually a plastic container with a remote control inside for weight) about six feet forward and three feet left from my bot's starting position. I've also placed my piece of dark paper to the right of the basket for the bot's Light Sensor to detect. If all goes well with my practice run, my bot should end up at Point D.

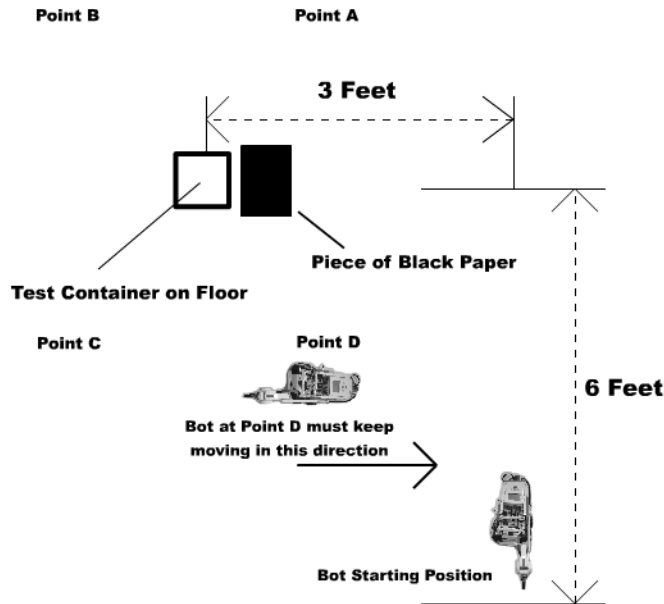


Figure 12-14. *Test environment for the SnapShotBot*

Because I've been testing my bot frequently, I wasn't surprised that my test run was successful. I knew from previous tests that my bot would enter the library, turn left, take the picture, and move forward until the Light Sensor was triggered. Getting the bot around the basket using a series of three left turns and forward movements worked perfectly. My bot ended up roughly in the area I've labeled Point D, pointed right and ready to finish its return.

Getting the Bot Home

The hard part is done. All that's left at this point is getting the bot back to you. There are numerous ways to do this, but the one I've selected uses the Sound Sensor again. If you'll look again at Figure 12-14, my bot is stopped at Point D, pointed to the right (back towards the dotted line where the bot first started its motion into the library). What I plan on doing is programming my bot to continue moving forward (moving to the right in Figure 12-14) until I "see" it. What I mean by "see" is that in the real library, the team will be looking through the hole in the wall. When the bot first enters the library and turns left towards the basket, the team will lose sight of the bot. Only when the bot returns and continues to move away from the basket will the team see the bot again. What I plan on doing is simply yelling "Stop!" The Sound Sensor will detect the noise, and I'll program the bot to stop, turn right, and come back to the hole in the wall. Simple!

First, you add a LOOP that will break when the Sound Sensor is triggered (see Figure 12-15).

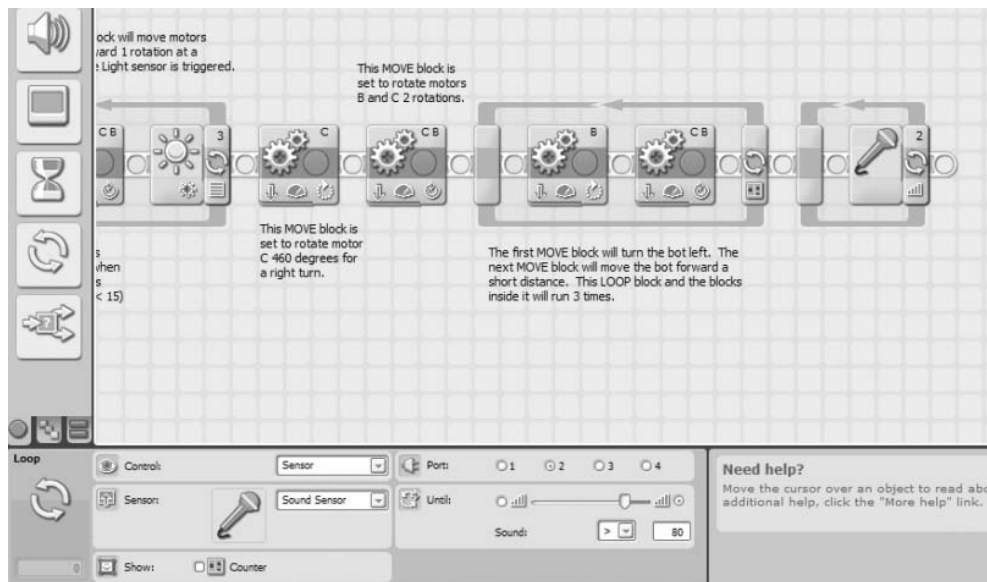


Figure 12-15. Add in a LOOP block with the Sound Sensor configured.

Next, you add in a MOVE block that will keep motors B and C moving until the Sound Sensor is triggered (see Figure 12-16).

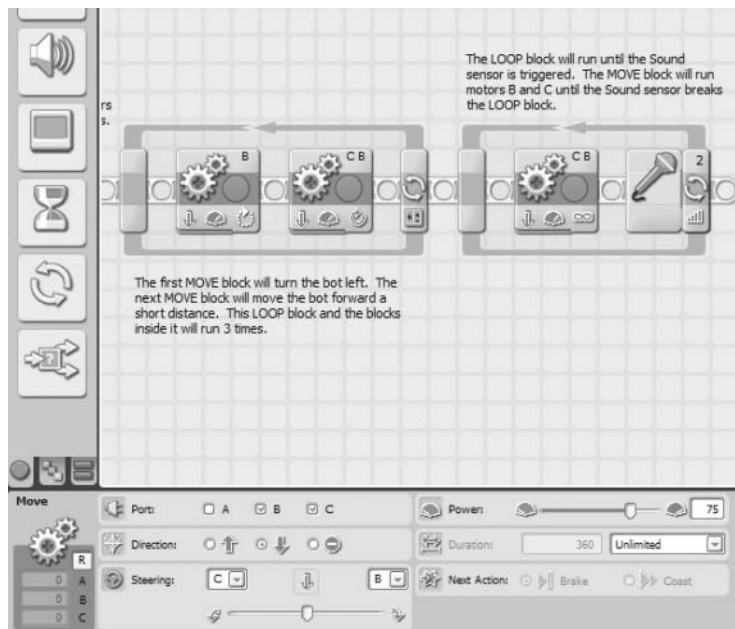


Figure 12-16. *The MOVE block will run the motors until the LOOP is broken by the Sound Sensor.*

Once again, we test. I had to adjust the Sound Sensor's sensitivity after my first run. I had it initially set to a value of 50, but I increased that to 80. I believe the sound of the motors spinning was loud enough to trigger the Sound Sensor, so once the bot started rolling, it immediately stopped. Only after I increased the value to 80 did the bot keep rolling until my voice (and loud “Stop!”) broke the LOOP block and the bot stopped.

The next task is “Turn right and move forward to library exit.” That should be easy enough. Drop in a MOVE block and configure it for a right turn (see Figure 12-17).

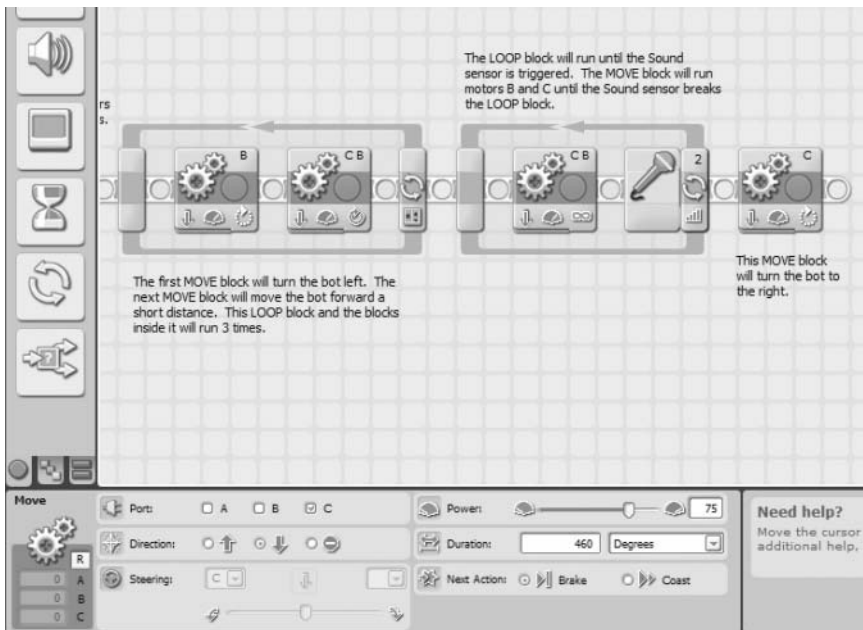


Figure 12-17. This MOVE block turns the bot right.

Use one final LOOP block to run motors B and C continuously. You'll simply reach in, grab the bot and turn it off . . . Oh, and get the picture developed. Drop in the LOOP block first (see Figure 12-18).

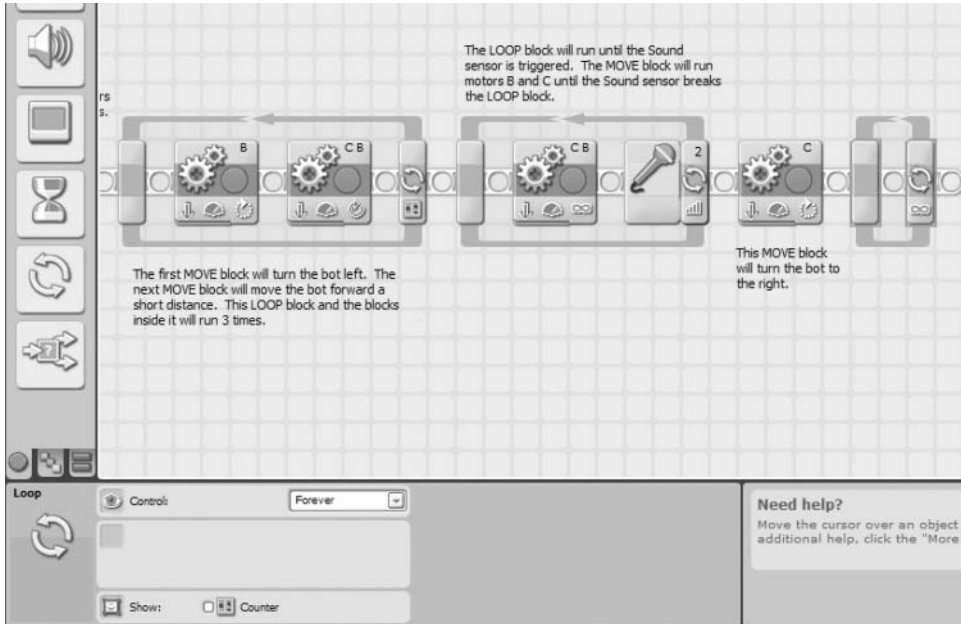


Figure 12-18. This LOOP block will run the motors continuously.

Then drop in the final MOVE block, configured to run motors B and C for an Unlimited duration (see Figure 12-19).

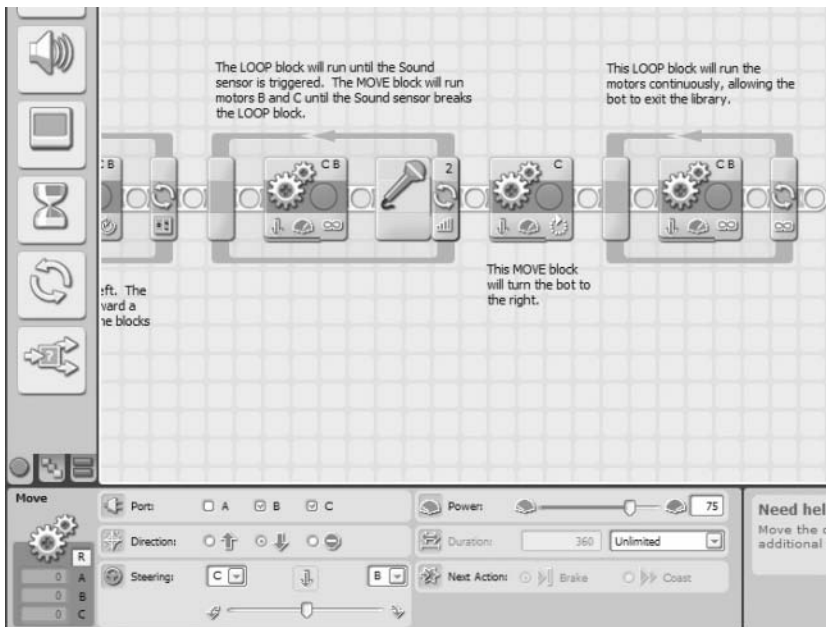


Figure 12-19. This MOVE block runs motors B and C.

That's it! Upload the program and test it. And test it again. Once you're confident that your SnapShotBot is running perfectly, set up your test environment one more time for the final test. Place the basket, the dark paper, and the bot in their starting positions. If you're using a regular camera or disposable camera, be sure you've wound the camera before sending the bot off. Place the arm that will press the camera's button directly above the button. Get two rolls of string (or twine) and tie one end of each roll to the SnapShotBot. I included two L-shaped beams on the back of my SnapShotBot where I could tie my strings. Remember to give plenty of slack on the string. Even better—have someone help you hold one of the rolls so you can both watch the strings and keep the tension off. When you're ready, press and release the Touch Sensor and watch your SnapShotBot begin its job. As it rolls in, be sure to feed out plenty of string—you don't want the string to get tight and pull the bot off course. When the bot turns to take the photograph, feed some more string as it moves towards the basket.

As the bot moves around the basket, you'll continue to feed the bot more string. Start watching for it to return, and when you see it, yell "*Stop!*" The bot should turn (don't forget to keep feeding more string) and head in your direction. If you've given the bot plenty of string, it should come right back to you. Grab the bot and turn it off.

Now, cut the two string ends from your bot and cut the string from the two rolls. Take all four string ends and place them together in your hand. Start pulling very slowly. Keep pulling slowly until you feel some resistance. When you feel the string tighten a little, pull even more slowly. At this point, the string should be pulling the basket towards you. Keep pulling until you've got the basket (and the key!) in your hands.

Congratulations! You've got the key and the team can continue exploring King Ixtua's tomb.